Haaga-Helia
ammattikorkeakoulu Oy

HTML5-mobiilisovelluksen kehittäminen AngularJS ja Cordova -tekniikoilla

Johannes Siipola

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2015

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

Päiväys

Tekijä(t)
Johannes Siipola

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
HTML5-mobiilisovelluksen kehittäminen AngularJS ja Cordova -tekniikoilla

Sivu- ja liitesivumäärä
24 + 2

Opinnäytetyön otsikko englanniksi
Developing a HTML5 mobile application using AngularJS and Cordova technologies

Tässä opinnäytetyössä käsitellään Popcult Companion-mobiilisovelluksen kehittämistä. Sovellus on suunnattu huhtikuussa 2015 järjestetyn Popcult Helsinki -populaarikulttuuritapahtuma kävijöille. Opinnäytetyössä käsitellään sovelluksen kehittämistä aina suunnittelusta sovelluskaupassa julkaisuun asti. Opinnäytetyössä käsitellään myös sovelluksesta saatua palautetta. Opinnäytetyön toimeksiantajana toimi tapahtuman järjestänyt yhdistys Finnish Fandom Conventions ry.

Popcult Companion -sovellus julkaistiin iOS- ja Android-alustoilla ja kehitettiin HTML5-hybriditekniikalla, joka tarkoittaa, että sovellus käyttää natiivien Objective-C/Swift- tai Java-tekniikoidan sijasta web-tekniikoita. Hybridisovellukset eivät ole ainoastaan verkkosivuja, vaan ne voivat hyödyntää puhelimen natiiveja toimintoja kuten kameraa tai ilmoituksia.

Sovelluksessa käytettiin hyödyksi monia erilaisia HTML5-hybridikehitykseen suunniteltuja sovelluskehyksiä kuten Cordova, AngularJS ja Ionic Framework. Opinnäytetyössä käsitellään näitä tekniikoita tarkemmin sekä pohditaan, mitä hyötyä niistä oli sovelluksen kehityksessä.

Opinnäytetyön lopuksi pohditaan sitä, soveltuivatko kehitykseen valitut työkalut sovelluksen toteutukseen sekä sitä, täyttikö opinnäytetyön tuloksena valmistunut sovellus sille asetetut tavoitteet. Kokonaisuudessaan tästä opinnäytetyöstä on hyötyä niille tahoille, jotka harkitsevat HTML5-hybridikehitystä mobiilisovelluksen kehittämisessä.

Asiasanat
ohjelmistokehitys mobiilisovellukset verkko-ohjelmointi tapahtumat

HAAGA-HELIA
University of Applied Sciences

Abstract

Date

Author(s)
Johannes Siipola

Degree programme
Degree Programme in Business Information Technology

Report/thesis title
Developing a HTML5 mobile application using AngularJS and Cordova technologies

Number of pages and appendix pages
24 + 2

This thesis is about building a mobile application called Popcult Companion. This application was intended for visitors of Popcult Helsinki convention. The thesis covers building the application from its initial design to its release in application stores. Analysing the feedback given by users of the application is also a part of this thesis. This thesis was commissioned by Finnish Fandom Conventions ry, a non-profit association which organized the event.

The Popcult Companion application was released on iOS and Android platforms and it was developed with hybrid HTML5 technology. This means that the application uses web technologies instead of native Objective-C/Swift or Java languages. Hybrid applications are not only web sites but they can also make use of native functionalities of the mobile platforms such as device camera or notifications.

The application used a multitude of programming frameworks such as Cordova, AngularJS, and Ionic Framework. This thesis covers these frameworks extensively and explains how they were used to build the application.

The final part of the thesis analyzes whether the chosen technologies were a good fit for the project and whether the application fulfilled its objectives. As a whole, this thesis is a resource for project teams or individuals interested in hybrid mobile development.

Keywords
software development, mobile applications, web programming, events

Sisällys

1 Johdanto

Popcult Helsinki oli Helsingin Kulttuuritalolla huhtikuussa 2015 ensimmäistä kertaa järjestettävä tapahtuma, jonka järjestäjänä toimi Finnish Fandom Conventions ry. Tapahtuma keskittyi populaarikulttuuriin kuten elokuviin, tv-sarjoihin, videopeleihin sekä sarjakuviin. Tapahtuma tarjosi kävijöilleen monipuolista ohjelmaa kuten luentoja, keskustelupaneeleita, esityksiä sekä työpajoja. Tapahtuman kohderyhmä olivat nuoret sekä nuoret aikuiset.

Olin itse mukana tapahtuman järjestämisessä. Tapahtumaa suunnitellessa syntyi idea siitä, että tapahtumalla voisi olla oma mobiilisovellus, jota tapahtuman kävijät voisivat käyttää tapahtuman aikana. Mobiilisovellus toisi lisäarvoa kävijälle, sillä se on kätevä tapa tutustua tapahtuman tarjoamaan ohjelmaan. Käyttäjät voivat merkitä sovellukseen omat suosikkinsa. Suosikkitoiminnon avulla käyttäjä voi myös saada muistutuksen kun hänen lempiohjelmansa alkaa. Sovellus on siis kätevä lisä tapahtuman tiedotukseen niille kävijöille, jotka haluavat hyödyntää älypuhelintaan tapahtuman aikana. Finnish Fandom Conventions ry toimi opinnäytetyön toimeksiantajana.

Koska sovellus oli tarkoitus saada käyttäjien saataville sekä iPhone- että Android-alustoille, valikoitui toteutustekniikaksi HTML5-hybridisovellus. Hybridisovellus on mobiilisovellus, joka on kirjoitettu web-tekniikoita kuten HTML-, CSS- ja Javascript-kieliä käyttäen, mutta sovellus voi kuitenkin käyttää natiiveja rajapintoja kuten esimerkiksi puhelimen kameraa tai ilmoituksia. Sekä iOS- että Android-alustat voivat hyödyntää samaa koodipohjaa.

Opinnäytetyössä käydään läpi projektin koko prosessi aina suunnittelusta julkaisuun asti. Samalla käydään läpi toteutuksen teknisiä yksityiskohtia. Esimerkiksi projektissa käytetyt kirjastot Cordova, Angular ja Ionic Framework käydään läpi. Tavoitteena on, että opinnäytetyöstä olisi hyötyä esimerkiksi sellaisille tahoille, jotka harkitsevat mobiilisovelluksen kehittämistä HTML5-hybriditekniikalla.

1.1 Käsitteet

HTML (Hypertext Markup Language): Kuvauskieli, jonka avulla voi rakentaa verkkosivuja tai muita dokumentteja. HTML5 on kielen viimeisin versio.

SDK (Software Development Kit): Sovellus, jolla voi kirjoittaa ohjelmistoja jollekkin alustalle. Jokaisella mobiilialustalla on oma SDK.

Natiivi sovellus: Mobiilisovellus, joka on kirjoitettu mobiilialustan omalla SDK:lla ja alustan käyttämällä ohjelmointikielellä.

Hybridisovellus: Mobiilisovellus, joka on kirjoitettu web-ohjelmointikieliä käyttäen. Sovellus voi silti hyödyntää puhelimen natiiveja toimintoja.

HTTP (Hypertext Transfer Protocol): Sovellusprotokolla, jonka avulla tietoa voi siirtää internetin yli.

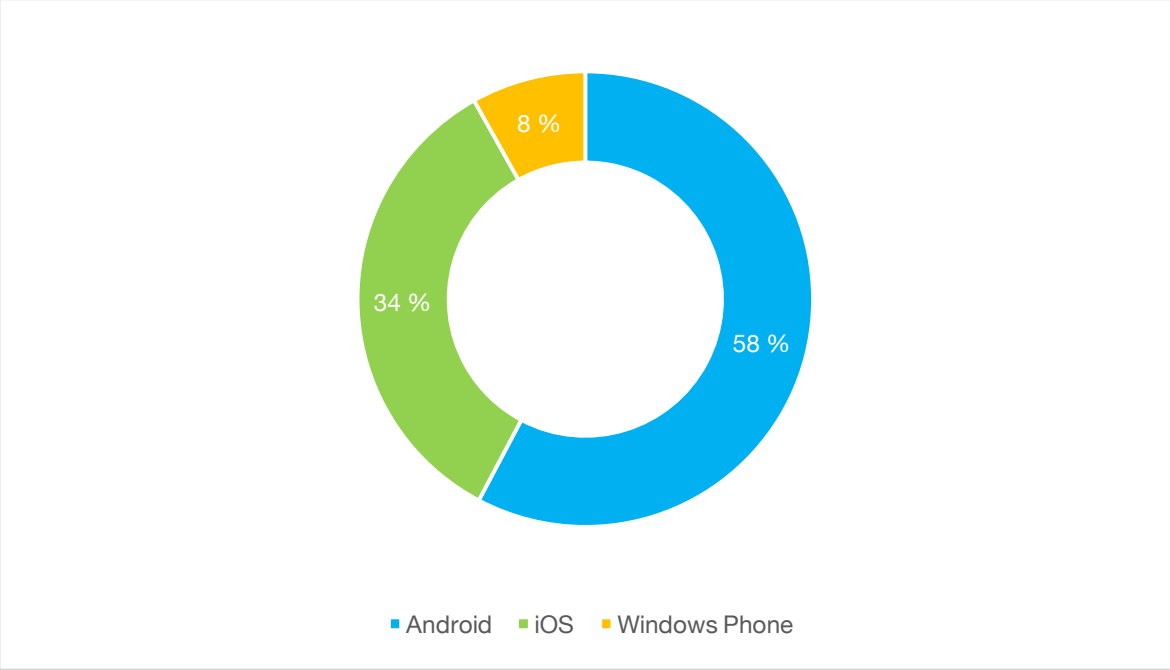
REST (Representational State Transfer): Arkkitehtuurimalli, jonka avulla voi toteuttaa web-rajapintoja, jotka perustuvat HTTP-kutsuihin. REST kuvaa minkälaisista osoitteista ja HTTP-kutsuista rajapinta koostuu.

2 Sovelluksen vaatimukset ja toiminnot

Popcult Companion on suunnattu Popcult Helsinki -tapahtuman kävijöille. Tämän takia on tärkeää, että sovelluksesta on hyötyä sen käyttäjälle. Tässä kappeleessa käydään läpi, minkälaiset käyttäjän vaatimukset sovellus täyttää ja minkälaisia toiminnallisuuksia vaatimukset pitävät sisällään.

2.1 Mobiilialustat

Jotta mobiilisovellus saataisiin kävijöiden käyttöön, täytyy varmistaa, että se on saatavilla niille mobiilialustoille, joita kävijät käyttävät. Maailmanlaajuisesti suosituin mobiilialusta on Googlen Android. Toisiksi suosituin alusta on Applen iOS. Windows Phone tulee kolmantena (IDC 2014). Suomen tilastot peilaavat maailmanlaajuisia tilastoja. (StatCounter 2015) Tarkimmat tiedot Popcult Helsinki -tapahtuman kävijöistä saadaan kuitenkin tapahtuman nettisivuston kävijätilastoja seuraamalla. Tapahtuman verkkosivun Google Analytics-tilistä poimittu data noudattaa samaa kaavaa kuin aiemmin mainitut tilastot. Windows Phone-käyttäjiä on vain 8% käyttäjistä. (Kuva 1)



Kuva 1. Popcult.fi -sivuston mobiilikävijät tammi-helmikuussa 2015

Koska kehityksessä ensisijaisesti käytetty Ionic Framework -viitekehys ei tukenut Windows Phone -alustaa, ja koska vain alle 10% potentiaalisista kävijöistä käytti Windows-alustaa, ei alustalle kehittäminen ollut aikataulu- ja resurssisyistä mielekästä. Sovellus kehitettiin iOS- ja Android-alustoille, koska näin saatiin katettua yli 90%

potentiaalista käyttäjistä. Windows Phone -versiota ei siis tämän projektin puitteissa toteutettu.

2.2 Käyttötapaukset

Popcult Companion -mobiilisovelluksen päätehtävä on palvella tapahtuman kävijää. Sovelluksen käyttötapaukset valittiin sen mukaan, minkä toimintojen koettiin tuovat eniten lisäarvoa potentiaaliselle käyttäjälle eli tapahtuman kävijälle.

Koska sovelluksen suunnitteluun ja toteuttamiseen oli hyvin rajallisesti aikaa, toteutettavaksi valittiin ne ominaisuudet, jotka ovat kaikkien tärkeimpiä. Lopullinen sovellus oli siis MVP eli Minimum Viable Product. Tämä tarkoittaa, että sovelluksessa pienin määrä ominaisuuksia, joilla siitä saadaan toimiva kokonaisuus. MVP:n avulla voidaan testata uuden sovellus- tai tuoteidean toimivuutta oikeilla käyttäjillä (Janssen).

2.2.1 Käyttäjä haluaa tietää mitä ohjelmia tapahtumassa on tarjolla

Sovelluksen päätoiminnallisuus on tapahtuman ohjelman listaaminen. Käyttäjä voi selata läpi listaa tapahtuman ohjelmanumeroiden nimistä sekä alkamis- ja päättymisajoista. Kun käyttäjä löytää mielenkiintoiselta kuulostavan ohjelman, hän voi saada siitä lisätietoja ohjelman nimeä painamalla. Näitä lisätietoja ovat esimerkiksi ohjelman pitäjä ja kuvaus.

2.2.2 Käyttäjä haluaa merkitä sovellukseen suosikkiohjelmansa

Jokainen kävijä nauttii tapahtumasta eri tavalla. On siis tärkeää, että käyttäjä pystyy mukauttamaan sovellusta mieleisekseen. Käyttäjä voi poimia ohjelmalistasta ne ohjelmat, jotka kiinnostavat häntä itseään eniten ja luoda näistä suosikkilistan. Jos käyttäjä on tehnyt suosikkilistan tapahtuman verkkosivustolla, hän voi tuoda suosikkinsa sovellukseen kirjautumalla sisään.

2.2.3 Käyttäjä haluaa muistutuksen, kun hänen suosikkiohjelmansa alkaa

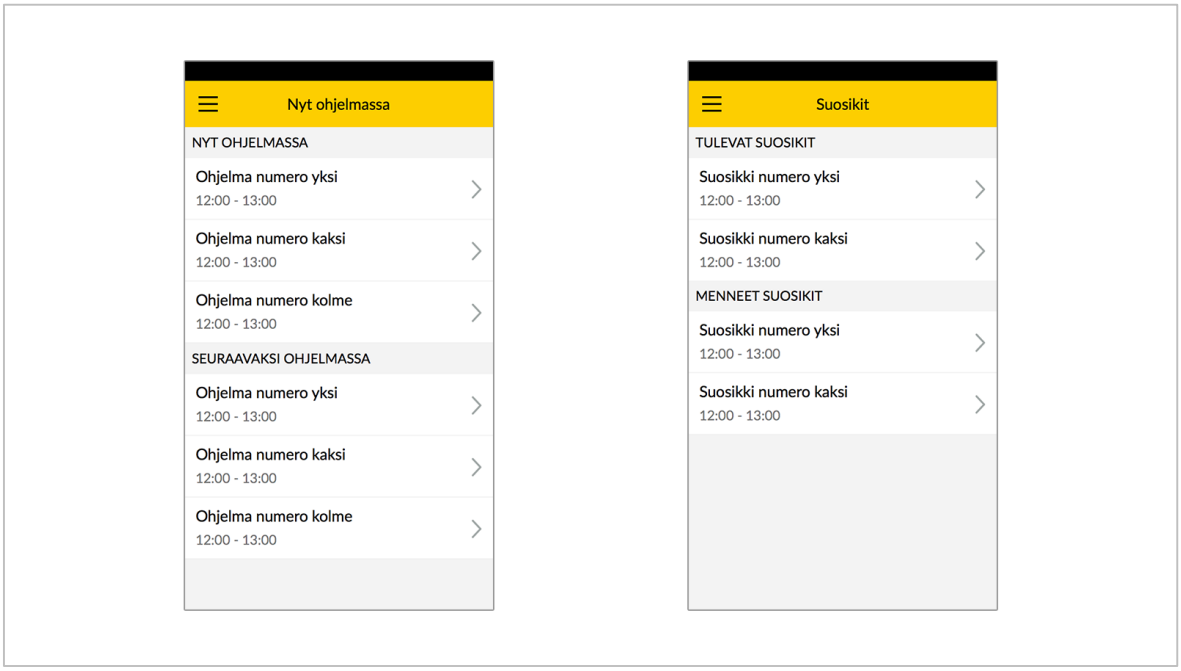
Sovellus tuo lisäarvoa käyttäjälle lähettämällä käyttäjälle ilmoituksen, kun käyttäjän suosikkiohjelma on alkamassa. Muistutus lähetetään käyttäjälle 15 minuuttia ennen ohjelman alkamista.

2.3 Käyttöliittymän suunnittelu

Popcult Helsinki sovelluksen käyttöliittymä suunniteltiin käyttäen mockup-tekniikkaa. Tämä tarkoittaa sitä, että sovelluksen näkymät suunniteltiin ja piirrettiin grafiikkaeditorilla ennen ohjelmointia.

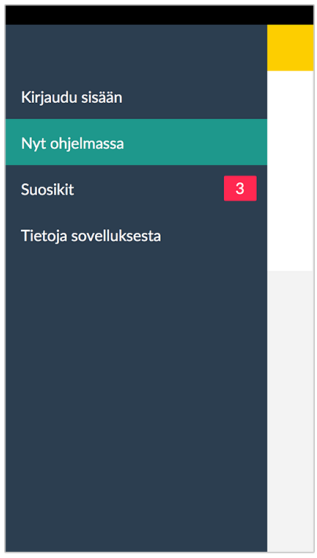
Sovelluksen käyttöliittymä on identtinen iOS- ja Android-laitteilla. Käyttöliittymä noudattelee parhaan mukaan iOS:n ja Androidin käyttöliittymäsuosituksia. Sovellus ei kuitenkaan jäljittele kummankaan alustan elementtejä sataprosenttisesti. Näin estetään niin sanottu uncanny valley - ilmiö. Tämä ilmiö tapahtuu, kun web-pohjainen käyttöliittymä yrittää jäljitellä natiivia käyttöliittymää tarkasti. Vaikka käyttöliittymä näyttää samalta, se ei kuitenkaan teknisistä rajoitteista johtuen kuitenkaan usein toimi täysin samalla tavalla kuin natiivi. Tämä rikkoo käyttäjän odotukset siitä, miten käyttöliittymä toimii ja vaikuttaa negatiivisesti käyttökokemukseen. (Higgins 2007)

Popcult Companionin käyttöliittymä ottaa vaikutteita sekä Androidin ja iOS:n käyttöliittymistä, mutta on kuitenkin hyvin omaleimainen sekä tapahtuman graafiseen ilmeeseen sopiva.



Kuva 2. Sovelluksen päänäkymät

Popcult Companion rakentuu kahden päänäkymän varaan. Toinen niistä on ”Nyt ohjelmassa” ja toinen ”Suosikit”. ”Nyt ohjelmassa” -näkymä listaa tällä hetkellä tapahtumassa käynnissä olevat ohjelmanumerot sekä niistä seuraavat ohjelmat. (Kuva 2) Näiden päänäkymien lisäksi sovelluksessa on ”Kirjaudu sisään” -näkymä.



Kuva 3. Sivuvalikko

Eri näkymien välillä pystyy liikkumaan niin sanotun sivuvalikon avulla. Sivuvalikko aukeaa painamalla kolmen päällekkäisen viivan muodostamaa ”hampurilaiskuvaketta”. Valikko listaa kaikki sovelluksen päänäkymät, joita painamalla voi siirtyä valittuun näkymään. (Kuva 3)

Kuva 4. Yksittäisen ohjelman tiedot

Yksittäistä ohjelmaa painamalla voi tarkastella sen tarkempia tietoja kuten ohjelman pitäjää, sijaintia sekä kuvausta. Ohjelman tiedot -sivulla olevaa tähtikuvaketta painamalla ohjelman voi lisätä suosikkeihin. (Kuva 4) Navigaatiopalkissa olevaa nuolikuvaketta voi palata edelliseen näkymään.

3 Sovelluksessa käytetyt tekniikat

Sovelluksen tekniikaksi valikoitui HTML5-hybriditekniikka. Hybridisovellus tarkoittaa sitä, että ohjelman käyttöliittymä ja toiminnallisuudet on toteutettu web-tekniikoita käyttäen, mutta sovellus voi silti käyttää natiiveja rajapintoja. Syyn tekniikan valintaan oli se, että HTML-, CSS- ja JavaScript-tekniikat olivat minulle tuttuja. Hybridisovellus oli vaihtoehdoista se, jolla sovelluksen sai kaikista nopeiten käyttäjien saataville useille eri alustoille.

Natiivit Android-ohjelmat kirjoitetaan Java-kielellä Googlen Android-rajapintoja käyttäen. Natiivi iOS-kehitys taas käyttää Swift ja Objective-C -kieliä sekä Applen rajapintoja. Hybridisovellusta kehittäessä ei tarvitse hallita kahta eri ohjelmointikieltä ja kahta kokoelmaa rajapintoja, vaan yksi koodipohja riittää kummallekin alustalle.

Täytyy huomioida, että hybridisovellusten kehittäminen ei sovellu kaikkiin sovelluksiin, esimerkiksi paljon laskentatehoa vaativiin peleihin. Popcult Companion on yksinkertainen listoihin ja valikoihin perustuva sovellus, eli hybriditekniikka soveltuu erinomaisesti sen toteuttamiseen. (Bristowe 2015)

Hybridisovellusten toteuttamiseen on olemassa useita erilaisia ohjelmistokehyksiä. Niistä yksi tunnetuimmista on Apache Cordova. Muita vaihtoehtoja ovat esimerkiksi AppGyver Steroids ja Appcelerator. Cordova valikoitui tekniikaksi sen takia, että se on avoimen lähdekoodin versio Adoben PhoneGapista, joka on tunnetuin ja yksi suosituimmista sovelluskehyksistä hybridikehittämiseen (Cowart 2013). Tämän takia sen käyttöön on saatavissa hyvin materiaalia internetistä.

3.1 Apache Cordova

Apache Cordova on viitekehys, jonka avulla voi kehittää mobiilisovelluksia web-tekniikoita hyödyntäen. Nämä sovellukset voi julkaista iPhone, Android ja Windows Phone -mobiilialustoilla. Cordova tarjoaa web-sovelluksille rajapinnan, jonka avulla ne voivat käyttää samoja toimintoja kuin natiivit sovellukset, esimerkiksi ottaa kuvia laitteen kameralla tai lähettää käyttäjälle ilmoituksia. (Apache 2015) Cordova on avoimen lähdekoodin versio Adoben PhoneGap-tuotteesta. Se on ominaisuuksilta käytännössä täysin identtinen koodiltaan verrattuna PhoneGapiin, mutta siitä puuttuu joitakin Adoben lisäpalveluita. (PhoneGap 2012)

3.2 Ionic Framework

Ionic Framework on kokoelma komponentteja ja kirjastoja, joiden avulla HTML5-pohjaisen mobiilisovelluksen kehitys on nopeampaa. Cordovassa itsessään ei tule minkäänlaista käyttöliittymäpohjaa mukana. Ionic tarjoaa hyvän pohjan mobiilisovellukselle: jokainen sovelluksen sivu on oma näkymänsä, joiden välillä voi navigoida. Ionicissa tulee mukana laaja valikoima erilaisia valmiita käyttöliittymäkomponentteja kuten painikkeita, valikoita ja listoja. Nämä komponentit ovat helposti kehittäjän muokattavissa. Ionic Framework perustuu Apachen Cordova ja Googlen AngularJS -viitekehyksiin, ja tukee iOS- ja Android-alustoja. (Ionic Framework 2015)

3.3 AngularJS

AngularJS on Googlen kehittämä JavaScript-viitekehys, joka helpottaa web-sovellusten kehittämistä. (AngularJS 2015a) AngularJS perustuu MVC-malliin, joka määrittelee koodille rakenteen: sovellus jaetaan kontrollereihin, palveluihin ja näkymiin, jotka keskustelevat keskenään.

Jokainen sovelluksen sivu on yksi näkymä. Näkymä hakee datan kontrollerista ja näyttää sen jossain tietyssä muodossa sille määritellyn pohjan mukaisesti. AngularJS:n pohjat perustuvat HTML-kieleen, johon on lisätty AngularJS:n omia laajennuksia.

Kontrolleri on sovelluksen kulmakivi. Se hakee dataa yhdestä tai useammasta palvelusta ja antaa datan näkymälle näytettäväksi.

Palvelut toimivat datan lähteenä tai tarjoavat usealle kontrollerille yhtenäistä toiminnallisuutta. Esimerkiksi programService -palvelu voi tarjota getPrograms -metodin. Jos joku sovelluksen kontrolleri tarvitsee listan ohjelmista, se voi kutsua metodia. Useampi kontrolleri voi käyttää yhtä palvelua hyväkseen. (AngularJS 2015b)

4 Sovelluksen toteuttaminen

Tässä kappaleessa käsitellään sovelluksen kehitysympäristöä, sovelluksen toimintoja ja niiden toteuttamista. Tämän lisäksi käsitellään sitä, miten Ionic Frameworkin komponentit saatiin muokattua Popcult Helsingin graafiseen ilmeeseen soveltuviksi.

4.1 Kehitysympäristö

Toisin kuin iOS-kehityksessä, jossa kehitysympäristönä toimii Xcode, tai Android-kehityksessä, jossa kehitysympäristö on Android Studio tai Eclipse, Cordova ja Ionic Framework eivät vaadi mitään erityistä kehitysympäristöä. Kehitys kuitenkin vaatii käytettyjen alustojen SDK:n asennetuksi, että projekti voidaan kääntää kullekin halutulle mobiilialustalle.

Projektissa käytetyksi editoriksi valikoitui JetBrains-yrityksen WebStorm. WebStorm on web-kehitykseen tarkoitettu IDE, jossa on hyvä JavaScript-kielen analysointi. WebStorm tukee myös projektissa käytettyjä Cordova - ja AngularJS-viitekehyksiä lisäosien avulla.

Sekä Android- että iOS-kehitysympäristöjen mukana tulee emulaattori, jolla voi esikatsella sovellusta ilman, että se täytyy asentaa oikealle laitteelle. Emulaattori voidaan käynnistää “ionic emulate ios” tai “ionic emulate android” -komennolla riippuen halutusta alustasta. iOS-emulaattori toimi hyvin ja käyttökokemus oli lähes sama kuin puhelimella. Android-emulaattori osoittautui käytännössä liian hitaaksi, jotta sen käyttäminen olisi ollut miellyttävää. Vaikka vaihtoehtona kokeiltiin VirtualBox-emulaattoriin perustuvaan nopeampaa Genymotion-emulaattoria, sovelluksen ajaminen emulaattorissa oli todella hidasta. Parhaiten Android-sovellusta pystyi kokeilemaan itse laitteella.

Sovelluksia on emulaattorin lisäksi mahdollista kokeilla myös itse päätelaitteella. iOS-laitteilla tämä vaatii kehittäjälisenssin. Kun kehittäjälisenssi on hankittu ja lisätty Xcode-sovellukseen, sovelluksen voi asentaa iOS-laitteeseen yhdistämällä laitteen tietokoneeseen USB-kaapelilla ja suorittamalla projektikansiossa komennon ”ionic run ios”. Android-päätelaitteella testaaminen on ilmaista. Se vaatii ainoastaan, että halutun laitteen asetuksista on laitettu päälle USB-virheenkorjaus. Kun laite on kytketty tietokoneeseen USB-kaapelilla, voi sovelluksen ajaa laitteessa komennolla ”ionic run android”.

iOS-sovellusta testattaessa sovelluksen lokia voi tarkastella tietokoneella Safari-selaimen avulla. Konsolin saa auki Safarin Develop-valikosta kohdasta iPhone Simulator. Valikon

alta löytyy lista tällä hetkellä ajettavista sovelluksista, joissa on mahdollisuus virheenkorjaukseen. Sovelluksen valitsemalla pääsee tarkastelemaan sovelluksen JavaScript-konsolia, HTML-rakennetta sekä sovelluksen muistissa olevaa sisältöä.

Android-laitteilla sovelluksen lokia pääsee tarkastelemaan tietokoneen Google Chrome -selaimella. Tämä tapahtuu valitsemalla Chromen valikosta ”More tools” ja ”Inspect devices”. Chrome listaa kaikki laitteet, joissa virheenkorjaus on mahdollista. Valitsemalla laitteen pystyy tarkastelemaan samoja asioita kuin Safarin kehittäjäkonsolissa.

Ionic Frameworkissä on tuki sovelluksen esikatselemiselle tietokoneen selaimessa käyttämällä ionic serve -komentoa. Tämä säästää paljon aikaa kehittämisessä, kuin projektia ei tarvitse kääntää jokaisen koodiin tehdyn muutoksen jälkeen. Sovelluksen esikatseleminen selaimessa on kuitenkin rajoittunut ainoastaan perustoiminnallisuuksiin. Esimerkksi ilmoituksia tai natiiveja ponnahdusikkunoita ei voi testata selaimessa, vaan ne vaativat projektin kääntämisen ja testaamisen joko mobiililaitteella tai emulaattorissa.

4.2 Tapahtuman ohjelmatietojen haku ja käsittely

Tapahtuman ohjelmaa pääsee tarkastelemaan osoitteessa http://ohjelma.popcult.fi/. Käyttäjälle näkyvien sivujen lisäksi ohjelmasivuilla on sovelluksille tarkoitettu REST-rajapinta, jonka avulla ohjelmiin ja käyttäjiin liittyvään tietoon pääsee käsiksi HTTP-kutsuja käyttäen. Data siirretään sovelluksen ja rajapinnan välillä JSON-muodossa. JSON on kevyt formaatti datan siirtämiseen, joka perustuu JavaScript-olioiden rakenteeseen.

Koska ohjelmatietoja ei ole mitään syytä ladata joka kerta uudestaan, kun käyttäjä avaa sovelluksen, ne tallennetaan HTML5 Local Storageen. Local Storage on avain-arvo-pareihin perustuva tallennusmenetelmä, jolla tietoa voidaan tallentaa selaimen muistiin. Se eroaa kekseistä siten, että tallennetun tiedon määrää ei ole rajattu yhtä paljon kuin kekseissä. (Pilgrim)

Koska ohjelmatietoihin voi tulla muutoksia, sovelluksessa on mekanismi, joka päivittää tiedot tietyn määräajan välein hakemalla ne uudestaan REST-rajapinnasta.

Ohjelmatiedoista huolehtii AngularJS-palvelu programService. Se hakee ohjelmatiedot Local Storagesta, ja jos ohjelmatiedot ovat vanhentuneet, se hakee ne uudestaan rajapinnasta ja tallentaa päivittyneet tiedot Local Storageen.

Kun ohjelmatiedot on haettu sovellukseen, käyttäjä voi tarkastella listaa ohjelmista sekä yksittäisen ohjelman ohjelmatietoja. Näiden näkymien kontrollerit hakevat ohjelmatiedot programServicestä ja muotoilevat ne oikeaan muotoon.

4.3 Suosikit

Sovelluksen käyttäjä voi lisätä omat suosikkiohjelmansa suosikkilistaan. Tämä tapahtuu painamalla jokaisen ohjelman kohdalla olevaa tähtikuvaketta. Jos suosikiksi lisätty ohjelma on tulevaisuudessa, ajastetaan ilmoitus, joka lähetetään käyttäjälle ennen ohjelman alkua.

AngularJS-palvelu favoriteService huolehtii kaikista suosikkilistaan liittyvistä toiminnoista. FavoriteServicestä löytyy metodit suosikin lisäämiseen, poistamiseen sekä käyttäjän suosikkilistan hakemiseen.

Käyttäjä pääsee tarkastelemaan omia suosikkejaan sovelluksen ”Suosikit”-näkymässä. Näkymä hakee suosikkitiedot favoriteService-palvelusta ja näyttää käyttäjälle listan hänen lisäämistään suosikeista.

Popcult Helsinki -tapahtuman nettisivulla on myös mahdollista luoda oma käyttäjätili ja lisätä ohjelmia suosikkeihin. Jos sovelluksen käyttäjä on jo luonut suosikkilistan tapahtuman verkkosivuilla, hän voi kirjautua sovellukseen sisään ja tuoda suosikkiohjelmansa sovellukseen. Tämä tapahtuu navigoimalla sovelluksen ”Kirjaudu sisään”-näkymään. Kun käyttäjä on kirjoittanut käyttäjätunnuksensa ja salasanansa, sovellus tekee HTTP-kutsun ohjelmakartan rajapintaan. Jos käyttäjätunnus ja salasana ovat oikein, rajapinta palauttaa sovellukselle listan käyttäjän suosikeista. Sovellus käy suosikkilistan läpi, ja jos listassa on suosikkeja, joita ei vielä löydy sovelluksen suosikeista, ne lisätään sovelluksen suosikkilistaan.

4.4 Ilmoitukset

Mobiilisovelluksissa on yleisesti käytössä kahdenlaisia ilmoituksia: push-ilmoituksia ja paikallisia ilmoituksia. Push-ilmoitukset toimivat siten, että puhelin kuuntelee jatkuvasti push-ilmoituksia lähettävää palvelinta. Kun palvelin lähettää uuden ilmoituksen, käyttäjä saa sen heti. (Apple 2014) Monet sosiaaliset palvelut kuten Facebook ja Twitter käyttävät push-ilmoituksia ilmoittaakseen esimerkiksi siitä, kun käyttäjä saa uuden viestin.

Toinen ilmoitustyyppi on paikalliset ilmoitukset. Ne eivät vaadi palvelinta niiden lähettämiseen vaan puhelin ajastaa ne johonkin tulevaisuuden ajanhetkeen. (Apple 2014) Tämä on kätevämpää sellaisissa tapauksissa, kun tiedetään etukäteen, mihin aikaan ilmoitus toimitetaan käyttäjälle.

Paikalliset ilmoitukset sopivat paremmin Popcult Companionin toimintalogiikkaan. Ne eivät myöskään vaadi palvelininfrastruktuuria ilmoitusten toimittamiseen. Kun käyttäjä lisää uuden suosikkiohjelman, sovellus ajastaa uuden ilmoituksen 15 minuuttia ennen ohjelman alkamisajankohtaa.

Sovelluksen ilmoitukset käyttävät Cordovan Local Notification -liitännäistä. Liitännäinen ei tule mukana Cordovan perusasennuksessa, vaan se pitää ladata erikseen Cordovan liitännäisrekisteristä (Cordova Local-Notification Plugin 2015). Asentaminen tapahtuu komennolla ”cordova plugin add https://github.com/katzer/cordova-plugin-local-notifications”.

Kun liitännäinen on asennettu, voidaan ilmoituksia lisätä liitännäisen tarjoaman rajapinnan avulla. Alla esimerkki koodista, joka lisää uuden ilmoituksen 15 minuuttia ennen ohjelman alkamisaikaa.

```
cordova.plugins.notification.local.schedule({
    id: program.id,
    at: program.start_time.clone().subtract(15, 'minutes').toDate(),
    title: "Suosikkiohjelmasi on alkamassa",
    text: program.name + " alkaa 15 minuutin kuluttua",
    icon: "res://silma_notification",
    smallIcon: "res://silma_notification"
});
```

Sovelluksessa ilmotusten lisäämisestä ja peruuttamisesta on vastuussa AngularJS-palvelu notificationService. Se tarjoaa sovellukselle 2 metodia, ilmoituksen lisäämiseen ja peruuttamiseen.

4.5 Käyttöliittymän mukauttaminen

Ionic Frameworkissä tulee mukana kattavat oletustyylit. Ne piti kuitenkin muokata vastaamaan Popcult Helsingin visuaalista ilmettä, johon kuuluu esimerkiksi tietty värimaailma ja fontit.

4.5.1 Tyylitiedostot

Tyylitiedostot määrittävät sovelluksen ulkoasun kuten värit ja fontit. Ionic Frameworkin valmiiden komponenttien tyylit on kirjoitettu SCSS-syntaksia käyttäen. SCSS on Sass-kielen murre. (Sass 2015a) Sass on CSS-preprosessori, joka laajentaa CSS-kieltä monilla uusilla ominaisuuksilla, kuten muuttujilla ja funktioilla. SCSS-syntaksilla kirjoitetut tiedostot käännetään lopuksi selaimen ymmärtämäksi CSS-tiedostoksi. (Sass 2015b)

Ionicin Frameworkin oletuskirjasinkoko on 14 pikseliä. Jotta teksti erottuisi paremmin, Popcult Companion käyttää 16 pikselin kokoista kirjaisinta. Ionic Frameworkin kirjasinkoko perustuu Sass-muuttujaan nimeltä ”$font-size-base”. Jotta tämä voidaan ylikirjoittaa, lisätään ionic.app.scss -tiedostoon määritys ”$font-size-base: 16px”. Tämä skaalaa fonttikoon koko sovelluksessa. Fontin lisäksi muutoksia piti tehdä sovelluksen väreihin sekä elementteihin kuten painikkeisiin. Osa komponenteista esimerkiksi ohjelmalistan yksittäinen rivi piti toteuttaa kokonaan mukautettuna komponenttina

4.5.2 Typografia

Ionic Frameworkin oletusfontti määräytyy järjestelmän mukaan. iOS-laitteilla se on Applen tuotteiden oletusfontti Helvetica. Android-laitteilla se on taas Googlen Roboto. Popcult Helsingin markkinoinnissa käytetty fontti on Google Fontsista saatava ilmainen Lato. Sitä käytetään esimerkiksi tapahtuman verkkosivulla. Jotta käyttäjä saisi yhtenäisen kokemuksen, käytettiin myös Popcult Companion -sovelluksessa samaa kirjasinta.

Mobiilisovelluksessa ei ole mielekästä ladata Google Fontsin kirjasimia verkon yli, vaan fontit kannattaa upottaa sovellukseen. FontSquirrel-palvelun avulla voi generoida Google Web Fonts -palvelusta ladatusta fonttitiedostosta verkkokäyttöön soveltuvia webfontteja. (FontSquirrel 2010) FontSquirrel-palvelu generoi sekä tarvittavat webfont-tiedostot että CSS-koodin, jolla niitä voidaan käyttää web-sivulla tai -sovelluksessa. Lato-fontti lisättiin Popcult Companion -projektiin siirtämällä palvelun generoimat fontit mukaan Cordova-projektin /www/fonts -kansioon ja lisäämällä seuraava koodi projektin CSS-määrittelyihin.

```
@font-face {
  font-family: 'Lato';
  src: url('fonts/lato-regular-webfont.eot');
  src: url('fonts/lato-regular-webfont.eot?#iefix') format('embedded-opentype'),
  url('fonts/lato-regular-webfont.woff2') format('woff2'),
  url('fonts/lato-regular-webfont.woff') format('woff'),
  url('fonts/lato-regular-webfont.ttf') format('truetype'),
  url('fonts/lato-regular-webfont.svg#latoregular') format('svg');
  font-weight: normal;
  font-style: normal;
}
```

Koska Lato-fontti on rekisteröity SIL Open Font License -lisenssin alla, täytyi kyseinen lisenssi sisällyttää sovellukseen muodossa, jossa käyttäjä pystyy lukemaan sen. (Addey 2012) Käytännössä tämä tapahtui lisäämällä lisenssi sovelluksen ”tietoja sovelluksesta” -osioon.

4.5.3 Kuvakkeet

Mobiisovelluksen käyttöliittymää voi rikastaa käyttämällä kuvakkeita. Ionic Frameworkin mukana tulee Ionicons -niminen kokoelma ikonifontteja. Ikonifontti tavallinen nettisivulle upotettava fontti, mutta kirjainten sijaan se koostuu erilaisista kuvakkeista. Valmis ikonifontti ei ollut Popcult Companionin käyttöön soveltuva vaihtoehto, koska ikonifonttia käytettäessä täytyy tyytyä niihin kuvakkeisiin, jotka sattuvat tulemaan mukana kyseisessä fontissa. Jotta kuvakkeet olisivat yhtenäisiä tapahtuman verkkosivulla käytettyjen kuvakkeiden kanssa, Popcult Companionissa käytettiin ikonifontin sijasta SVG-muotoisia kuvakkeita.

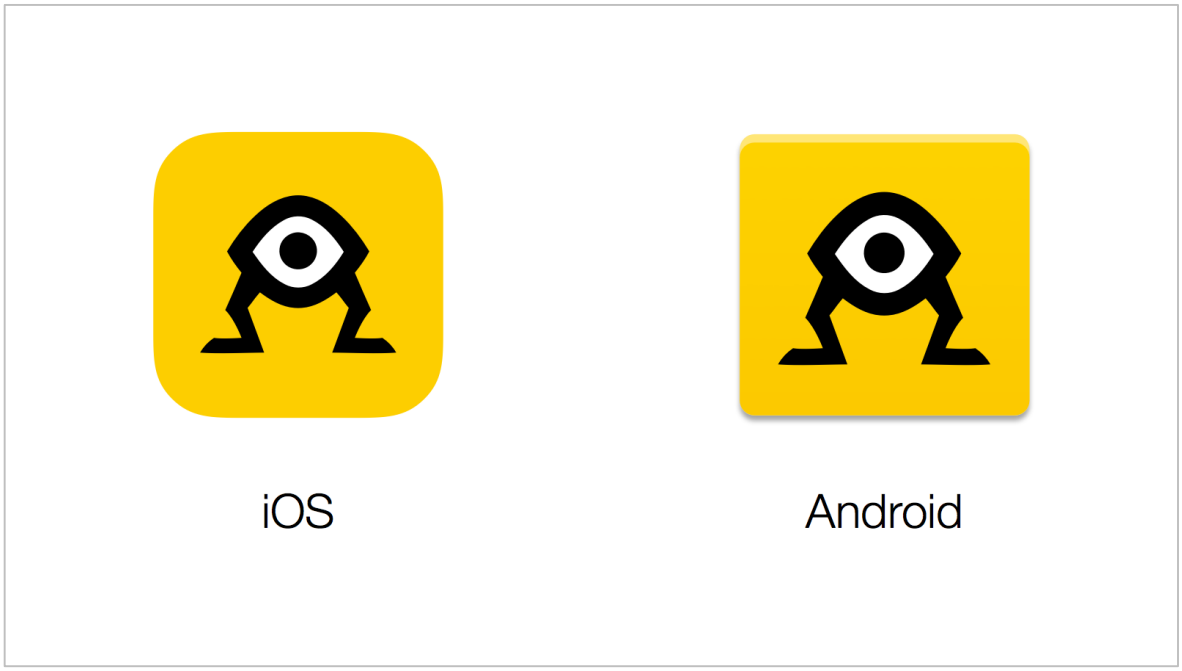
SVG-muotoiset kuvakkeet ovat vektorigrafiikkaa. Tämä tarkoittaa sitä, että niiden kokoa voi muuttaa ilman, että kuvakkeen laatu kärsii. Tämä on tärkeää sen takia, että sovelluksen täytyy toimia laitteilla, joiden näytössä on eri pikselitiheyksiä. Vektorikuvaketta käytettäessä riittää, että jokaista kuvaketta kohden on vain yksi vektorikuva.

iOS-alustoilla pikselitiheyksiä on 3 erilaista. 1x on se, mitä alkuperäinen iPhone ja iPad käyttivät. Retina-näytöllä varustetut iPhonet ja iPadit käyttää 2x-pikselitiheyttä. iPhone 6 Plus käyttää Retina HD -näyttöä, jonka pikselitiheys on 3x. (Apple 2015d)

Vektorikuvakkeiden käyttäminen on kuitenkin tärkeämpää Android-alustalla, sillä Android-laitteiden pikselitiheydet vaihtelevat suuresti. Ne on jaoteltu kuuteen eri kategoriaan (Google 2015b):

- LDPI (0.75x)
- MDPI (1x)
- HDPI (1.5x)
- XHDPI (2x)
- XXHDPI (3x)
- XXXHDPI (4x)

Sovelluksen käyttöliittymän kuvakkeiden lisäksi sekä iOS että Android vaativat sovellukselle kuvakkeen, joka näkyy laitteen kotinäkymässä tai sovelluslistassa. Vaikka käyttöliittymässä käytetyt kuvakkeet voivat olla SVG-muotoisia, sekä iOS- että Android-alustojen aloitusruutujen kuvakkeiden pitää olla PNG-muotoisia.



Kuva 5. Sovelluksen kuvakkeet iOS- ja Android-alustoilla.

Popcult Companion kuvakkeena toimi tapahtuman maskotti Silmä. Kuvakkeesta suunniteltiin kaksi versiota, yksi iOS-alustalle ja yksi Android-alustalle. (Kuva 5) iOS-versio on pelkästään keltaisella pohjalla oleva Silmä ilman läpinäkyvyyttä, koska iOS lisää automaattisesti kuvakkeeseen pyöreät reunat. Android-versiossa kuvakkeeseen täytyi sisällyttää läpinäkyvä tausta sekä varjo, koska Android käyttää kuvaketta sellaisenaan. Varjo on tärkeä, jotta kuvake erottuu eri käyttäjien erilaisista taustakuvista.

Kuvakkeet suunniteltiin vektorimuodossa. Kuvakkeiden suunnittelun jälkeen niistä piti tallentaa rasteriversiot kummankin käyttöjärjestelmän vaatimien pikselikokojen mukaisesti. Näiden lisäksi kuvakkeista tarvittiin korkealaatuiset versiot App Storen ja Play Storen sovelluslistausta varten.

5 Sovelluksen julkaisu sovelluskaupoissa

Jotta sovellus saadaan loppukäyttäjän saataville, se täytyy ensin julkaista sovelluskaupassa. Tässä kappaleessa kerrotaan, kuinka julkaisuprosessi toimii Applen App Storessa ja Googlen Play Storessa.

5.1 iOS-version julkaisu Apple App Storessa

Applen App Store on palvelu, josta voi ladata sovelluksia iPhone-, iPad- ja iPod Touch -laitteisiin. (Apple 2015a) Se on ainoa tapa, jolla loppukäyttäjät voivat saada sovelluksia Applen mobiililaitteisiin, jos ei lasketa mukaan Applen yrityskäyttöön suunnattua Enterprise App Storea. Enterprise App Store on suunniteltu yritysten sisäisten sovellusten jakeluun. (Apple 2015b)

5.1.1 Rekisteröityminen Apple Developer -ohjelmaan

Sovellusten julkaisuun App Storessa vaaditaan rekisteröityminen Apple Developer -ohjelmaan. Rekisteröityminen maksaa 99 euroa vuodessa. Jos maksua ei maksa vuosittain, kaikki julkaisijan ohjelmat poistetaan App Storesta. Rekisteröityminen tapahtuu Apple Developer -sivulla. Rekisteröityessään voi valita joko iOS Developer -ohjelman tai Mac Developer -ohjelman, jotka kummatkin maksavat 99 euroa per vuosi. Rekisteröitymisen jälkeen maksu suoritetaan iTunesin kautta.

5.1.2 Julkaisuversion luominen

App Storeen ladattavan julkaisuversion luomiseen käytetään Xcode -kehitysympäristöä. Xcoden Product-valikosta valitaan ”Archive”. Kun Xcode on validoinut sovelluksen, valitaan kehittäjälisenssi, jonka avulla julkaisuversio luodaan. Tämän jälkeen Xcode luo julkaisuun tarvittavan arkiston. Kun arkisto on valmis, se voidaan lähettää Applelle painamalla ”Submit”.

5.1.3 App Store -listaus

Sovellus lisätään App Storeen Applen iTunes Connect -sivuston kautta. iTunes Connect -palveluun kirjautumisen jälkeen valitaan valikosta ”My Apps”. Tämän jälkeen painetaan pluskuvaketta ja valitaan ”New iOS App”. Tämän jälkeen syötetään tarvittavat tiedot, joita ovat sovelluksen nimi ”Popcult Companion”, sovelluksen versio ”1.0” ja sovelluksen ensisijainen kieli ”Finnish”. Tämän jälkeen valitaan sovelluksen Bundle ID eli kehittäjän tunniste. Valitaan ”Xcode iOS Wildcard App ID”. Tämän jälkeen valitaan sovelluksen Bundle ID suffix eli sovelluksen tunniste. Tunnisteeksi suositellaan reverse domain -

tyyppistä tunnistetta, joka koostuu kehittäjän URL-osoitteesta sekä sovelluksen nimestä. Popcult Companionin tapauksessa tunnisteeksi valitaan ”fi.popcult.companion”.

Kun sovelluksen perustiedot on täytetty, päästään täyttämään sovelluksen App Store -sivulla näkyviä tietoja. Vaadittavia graafisia elementtejä ovat korkealaatuinen sovelluksen kuvake sekä vähintään yksi kuvankaappaus jokaiselta iOS-laitteen versiolta, jota sovellus tukee. (Apple 2015c) Popcult Companion -sovelluksessa nämä olivat:

- 3.5 tuumaa (iPhone 4)
- 4 tuumaa (iPhone 5)
- 4.7 tuuma (iPhone 6)
- 5.5 tuumaa (iPhone 6 Plus)
- 9.7 tuumaa (iPad)

Kun kuvankaappaukset on ladattu iTunes Connectiin, voidaan täyttää loput tarvittavat tiedot. Näitä ovat:

- Sovelluksen nimi
- Sovelluksen kuvaus
- Avainsanat
- Osoite sovelluksen tukisivustolle
- Osoite sovelluksen tietosuojalausuntoon
- Sovelluksen tekijä
- Sovelluksen versio
- Sovelluksen kategoria

Kun tiedot on täytetty, voidaan tämän jälkeen valita sovelluksen ikäraja sekä hinta.

Sen jälkeen, kun sovellusarkisto on ladattu Xcoden avulla iTunes Connectiin, pitää vielä täyttää tarvittavat tiedot App Storen arviointiprosessia varten. Näitä ovat yhteyshenkilön nimi, puhelinnumero sekä sähköpostiosoite. Tämän lisäksi koska sovelluksessa on sisäänkirjautumistoiminto, pitää täyttää testitunnuksen käyttäjänimi, salasana sekä sisäänkirjautumiseen liittyvät muistiinpanot.

Kun listaus on valmis, voidaan sovellus lähettää Applen arvioitavaksi. Arvioinnissa kestää noin viikko. Kun arviointi on valmis ja sovellus on hyväksytty, sovellus ilmestyy ladattavaksi App Storeen muutaman tunnin jälkeen.

5.2 Android-version julkaisu Google Play Storessa

Googlen virallinen sovellusten jakelukanava on Google Play Store, jonka kautta voi ladata sovelluksia Android-laitteisiin. (Google 2015a) Android-laitteisiin on myös mahdollista asentaa sovelluksia Play Storen ohi, mutta tämä vaatii, että laitteen asetuksista on laitettu päälle tuntemattomien sovelluslähteiden hyväksyminen. Tämän jälkeen verkosta voi ladata ja asentaa APK-muotoisen sovellustiedoston. (CNET, 2013) Tämä ei ole kuitenkaan peruskäyttäjälle suunnattu prosessi.

5.2.1 Rekisteröityminen Google Play-kehittäjäohjelmaan

Jotta sovelluksia voi julkaista Google Play -kaupassa, täytyy ensin rekisteröidä kehittäjätili. Tämä vaatii 15 euron kertamaksun.

5.2.2 Julkaisuversion luominen

Android-julkaisuversio luodaan Ionic Frameworkin konsolikomennolla ”cordova build --release android”.

Komento luo tiedoston “CordovaApp-release-unsigned.apk”. Jotta tiedosto voidaan julkaista Google Play Storessa, se täytyy ensin allekirjoittaa. Allekirjoittamista varten pitää ensin generoida salausavain komennolla:

“keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 10000”

Tämän jälkeen syötetään avaimeen liitettävä salasana sekä muut tarvittavat tiedot, joihin kuuluu esimerkiksi salausavaimen haltijan nimi sekä yhteystiedot. Kun salausavain on generoitu, se löytyy kansiosta nimellä ”my-release-key.keystore”. Seuraavaksi allekirjoitetaan asennuspaketti komennolla:

“jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore CordovaApp-release-unsigned.apk alias_name”

Allekirjoittamisen jälkeen asennuspaketti pitää optimoida zipalign-työkalulla. Tämä tapahtuu komennolla

“zipalign -v 4 CordovaApp-release-unsigned.apk PopcultCompanion.apk”

Lopullinen asennuspaketti löytyy kansiosta nimellä PopcultCompanion.apk.

5.2.3 Play Store -listaus

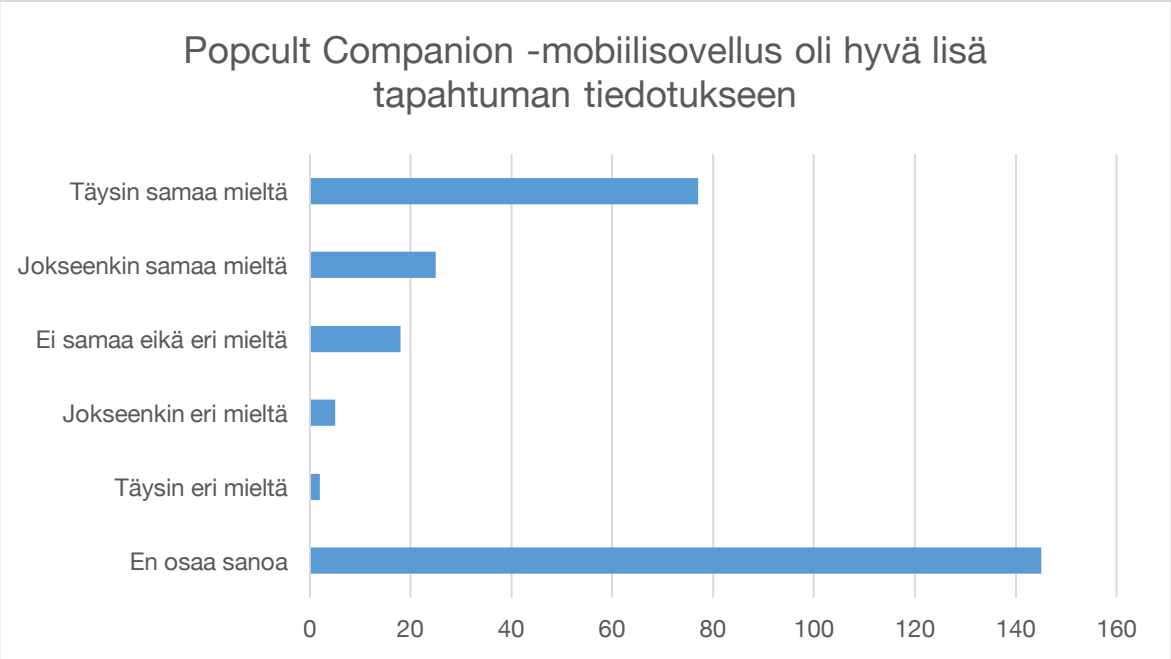
Ennen kuin sovellus voidaan ladata Play Storeen, sille täytyy luoda listaus. Tämä tapahtuu siirtymällä Google Play Developer Console -verkkosivustolle ja valitsemalla painike “Add New Application”. Tämän jälkeen valitaan sovelluksen oletuskieli, tässä tapauksessa ”Finnish”. Sitten kirjoitetaan sovelluksen nimi ”Popcult Companion”. Lopuksi valitaan ”Prepare Store Listing” ja täytetään tarvittavat Play Storen listauksessa näkyvät tiedot:

- Sovelluksen lyhyt kuvaus,
- Sovelluksen pitkä kuvaus
- Kaksi sovelluksen kuvankaappausta
- Korkealaatuinen sovelluksen kuvake
- Sovelluksen banneri
- Sovelluksen tyyppi: sovellus/peli
- Sovelluksen kategoria
- Kehittäjän sähköpostiosoite
- Linkki sovelluksen tietosuojalausuntoon

Sovelluksen kuvankaappauksiksi riittää 2 kuvankaappausta sovelluksen puhelinversiosta. Kun tarvittavat tiedot on täytetty, voidaan ladata sovelluksen APK-asennuspaketti Play Storeen valitsemalla ”Upload your first apk to production”. Tämän jälkeen valitaan aiemmin luotu AKP-tiedosto tietokoneelta ja ladataan se Play Storeen. Kun asennuspaketti on ladattu, voidaan valita sovelluksen ikäraja. Tämän jälkeen sovellus voidaan julkaista sovelluskaupassa painamalla ”Publish”-painiketta. Julkaisun jälkeen kestää muutama tunti, ennen kuin sovellus tulee ladattavaksi Play Storeen.

6 Sovelluksesta saatu palaute

Popcult Companionia ladattiin ennen tapahtumaa ja sen aikana yhteensä 225 eri puhelimelle ja tabletille. Näistä 88 oli iOS-laitteita ja 137 Android-laitteita. Tapahtumassa oli kävijöitä noin 1000. Tämä tarkoittaa, että keskimäärin viidennes kävijöistä latasi sovelluksen puhelimeensa.



Kuva 6. Ote Popcult Helsinki -tapahtuman kävijäpalautteen koosteesta

Popcult Helsinki -tapahtuman jälkeen avattiin tapahtuman kävijöille palautelomake, jonka avulla he pystyivät antamaan palautetta tapahtumasta. Yksi kävijäpalautteessa esitetty kysymys koski Popcult Companion -mobiilisovellusta. Kävijöitä pyydettiin arvioimaan sitä, oliko sovellus hyvä lisä tapahtuman tiedotukseen. Palautelomakkeen vastaajat pystyivät myös kommentoimaan sovellusta tapahtuman tiedottamista koskevassa vapaassa tekstikentässä. Enemmistö sovellusta käyttäneistä kävijöistä oli sitä mieltä, että sovellus oli hyvä lisä tapahtumaan (Kuva 6). Käyttäjät antoivat paljon positiivista palautetta ja kehitysehdotuksia sovelluksesta. Sovelluksen ylläpito ja kehittäminen on siis kannattavaa seuraavissakin Popcult Helsinki -tapahtumissa. Palautteen avulla on hyvä suunnitella sitä, minkälaisia päivityksiä ja uusia toimintoja sovellukseen voi toteuttaa seuraavaa tapahtumaa silmällä pitäen.

7 Pohdinta

Tärkeä kysymys sovelluksen kannalta on se, sopiko valittu toteuttamistekniikka eli HTML5-hybridikehitys Popcult Companion -sovelluksen kehittämiseen. Sopivatko valitut sovelluskehykset sovelluksen kehittämiseen?

HTML5-hybridikehitys osoittautui hyväksi valinnaksi sovelluksen kehitykseen. Sen avulla pystyin käyttämään hyväksi minulla jo valmiiksi olevia HTML-, CSS- ja JavaScript-taitoja sovelluksen kehittämisessä. Jos sovellus olisi toteutettu natiivina Android- ja iOS-sovelluksena, en olisi todennäköisesti yksin saanut sitä valmiiksi tapahtumaan mennessä, sillä minulla ei ole entuudestaan kokemusta Androidin Java-ohjelmoinnista tai iOS:n Objective-C ja Swift-ohjelmointikielistä. Lopullinen sovellus oli myös käyttökokemukseltaan miellyttävä, eikä valittu teknologia vaikuttanut negatiivisesti sovelluksen toimintaan.

Ionic Framework oli hyvä valinta viitekehykseksi. Vaikka Cordova onkin kattava kirjasto hybridisovellusten kirjoittamiseen, se toimi käytännössä ainoastaan selaimena, jonka avulla verkkosivuja voi ajaa kännykkäsovelluksina ja rajapintana natiiveihin toimintoihin. Cordova ei ottanut kantaa ollenkaan sovelluksen rakenteeseen tai käyttöliittymään. Ionic Framework paikkasi hyvin Cordovan puutteita ja antoi hyvin pohjan sovelluksen kehittämiselle.

On myös tärkeä arvioida sitä, onnistuiko sovellus tehtävässään, eli toiko sovellus lisäarvoa tapahtuman kävijöille. Sovellukseen toteutettiin aikarajoitteiden takia vain 3 käyttäjätarinaa, jotka olivat sovelluksen tärkeimmät toiminnot. Riittikö sovelluksessa toiminnallisuutta käyttäjien tarpeisiin?

Palautteen mukaan käyttäjät olivat hyvin tyytyväisiä sovellukseen. Palautteesta näkee, että sovellus oli kävijöiden mielestä hyödyllinen. Yhdessäkään palautteessa ei oltu sitä mieltä, että sovellus olisi ollut ominaisuuksiltaan puutteellinen. Popcult Companion siis onnistui siis tehtävässään ja toi lisäarvoa tapahtuman kävijälle. Palautteen ”vapaa sana”-osiosta saatiin myös kullanarvoisia kehitysehdotuksia, jotka voidaan ottaa huomioon sovellusta jatkokehitettäessä.

Popcult Companion on ensimmäinen mobiilisovellus, jota olen ollut kehittämässä. Vaikka sovelluksen käyttämät web-tekniikat olivat minulle tuttuja, kehittämiseen valitut sovellusviitekehykset olivat täysin uusia. Sovelluksen kehittäminen oli haastavaa tiukan aikataulun vuoksi, mutta lopputulos oli kuitenkin toimiva ja eheä kokonaisuus. Tässä auttoi

paljon sovelluksen toimintojen rajoittaminen ainoastaan kaikkein tärkeimpiin. Kokonaisuudessaan sovelluksen kehittäminen oli hyödyllinen oppimiskokemus varsinkin projektinhallinnollisesta näkökulmasta. Vaikka kehittämistyötä onkin tekemässä yksin, on silti tärkeää rajata kehityksen vaatimukset sellaisiksi, että ne pystyy projektin aikataulun puitteissa toteuttamaan.

Mobiilikehitys on kiinnostava ohjelmistokehityksen ala, jolla on omat haasteensa sekä tekniikoiden, että käytettävyyden suhteen. Sovelluksesta saatu positiivinen palaute inspiroi minua jatkokehittämään Popcult Companionia jatkossakin tulevia tapahtumia ajatellen. Koska sain projektin puitteissa hyviä kokemuksia mobiilikehityksestä, voisi olla kiinnostavaa päästä toteuttamaan tulevaisuudessa muitakin mobiilisovelluksia.

Lähteet

Addey, D. 2012. Free embeddable fonts for iOS apps. Luettavissa http://daveaddey.com/?p=916. Luettu 1.4.2015.

AngularJS 2015a. Luettavissa: https://angularjs.org Luettu 1.4.2015.

AngularJS 2015b. Developer Guide. Conceptual Overview. Luettavissa: https://docs.angularjs.org/guide/concepts Luettu 1.4.2015.

Apache 2015. Apache Cordova. Luettavissa: https://cordova.apache.org Luettu 1.4.2015.

Apple 2014. iOS Human Interface Guidelines.

Apple 2015a. iPhonen käyttöopas.

Apple 2015b. iOS Developer Enterprise Program. Luettavissa: https://developer.apple.com/programs/ios/enterprise/ Luettu 18.4.2015.

Apple 2015c. iTunes Connect Properties. Luettavissa https://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide/Appendices/Properties.html Luettu 18.4.2015.

Apple 2015d. Icon and Image Sizes. Luettavissa https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/IconMatrix.html Luettu 1.4.2015.

Bristowe J. 2015. What is a Hybrid Mobile App? Luettavissa: http://developer.telerik.com/featured/what-is-a-hybrid-mobile-app/ Luettu 1.4.2015.

CNET 2013. How to install apps outside of Google Play. Luettavissa: http://www.cnet.com/how-to/how-to-install-apps-outside-of-google-play/ Luettu 1.4.2015.

Cordova Local-Notification Plugin 2015. Installation. Luettavissa https://github.com/katzer/cordova-plugin-local-notifications/wiki/03.-Installation Luettu 1.4.2015.

Cowart, J. 2013. Pros and Cons of the Top 5 Cross-Platform Tools. Luettavissa: http://www.developereconomics.com/pros-cons-top-5-cross-platform-tools/ Luettu 17.5.2015 .

FontSquirrel 2010. The Official "How To Use The Generator" Post. Luettavissa: http://www.fontsquirrel.com/blog/2010/12/how-to-use-the-generator Luettu 1.4.2015.

Google 2015a. The Google Play Opportunity. Luettavissa https://developer.android.com/distribute/googleplay/about.html Luettu 18.4.2015.

Google 2015b. Supporting Multiple Screens. http://developer.android.com/guide/practices/screens_support.html Luettu 20.5.2015.

Higgins, B. 2007. The Uncanny Valley of user interface design. Luettavissa: http://billhiggins.us/blog/2007/05/17/the-uncanny-valley-of-user-interface-design/ Luettu: 17.5.2015.

IDC 2014. Smartphone OS Market Share Q4 2014. Luettavissa http://www.idc.com/prodserv/smartphone-os-market-share.jsp Luettu 1.4.2015.

Ionic Framework 2015. Luettavissa: http://ionicframework.com Luettu 1.4.2015.

Janssen, C. Technopedia, Minimum Viable Product (MVP). Luettavissa: http://www.techopedia.com/definition/27809/minimum-viable-product-mvp Luettu: 19.5.2015.

PhoneGap 2012. PhoneGap, Cordova, and what’s in a name? Luettavissa: http://phonegap.com/2012/03/19/phonegap-cordova-and-what’s-in-a-name/ Luettu 1.4.2015.

Pilgrim, M. Diving Into HTML5. Luettavissa http://diveintohtml5.info/storage.html Luettu: 20.4.2015.

Sass 2015a. Sass Documentation. Luettavissa: http://sass-lang.com/documentation/file.SASS_REFERENCE.html Luettu 1.4.2015.

Sass 2015b. Sass Basics. Luettavissa: http://sass-lang.com/guide Luettu 1.4.2015.

StatCounter 2015. Statcounter Global Stats. Luettavissa: http://gs.statcounter.com/#mobile_os-FI-monthly-201401-201502-bar Luettu 1.4.2015.